

Adaptive Sub-Linear Time Fourier Algorithms

David Lawlor, Yang Wang, and Andrew Christlieb
Department of Mathematics, Michigan State University

July 27, 2012

Abstract

We present a new deterministic algorithm for the sparse Fourier transform problem, in which we seek to identify $k \ll N$ significant Fourier coefficients from a signal of bandwidth N . Previous deterministic algorithms exhibit quadratic runtime scaling, while our algorithm scales linearly with k in the average case. Underlying our algorithm are a few simple observations relating the Fourier coefficients of time-shifted samples to unshifted samples of the input function. This allows us to detect when aliasing between two or more frequencies has occurred, as well as to determine the value of unaliased frequencies. We show that empirically our algorithm is orders of magnitude faster than competing algorithms.

1 Introduction

The Fast Fourier Transform (FFT) is arguably the most ubiquitous numerical algorithm in scientific computing. In addition to being named one of the “Top Ten Algorithms” of the past century [DS00], the FFT is a critical tool in myriad applications, ranging from signal processing to computational PDE and machine learning. At the time of its introduction, it represented a major leap forward in the size of problems that could be solved on available hardware, as it reduces the runtime complexity of computing the Discrete Fourier Transform (DFT) of a length- N array from $O(N^2)$ to $O(N \log N)$.

Any algorithm which computes all N Fourier coefficients has a runtime complexity of $\Omega(N)$, since it takes that much time merely to report the output. However, in many applications it is known that the DFT of the signal of interest is highly sparse – that is, only a small number of coefficients are non-zero. In this case it is possible to break the $\Omega(N)$ barrier by asking only for the largest k terms in the signal’s DFT. When $k \ll N$ existing algorithms can significantly outperform even highly optimized FFT implementations [IGS07, Iwe10, HIKP12b].

1.1 Related Work

The first works to implicitly address the sparse approximate DFT problem appeared in the theoretical computer science literature in the early 1990s. In [LMN93],

a variant of the Fourier transform for Boolean functions was shown to have applications for learnability. A polynomial-time algorithm to find large coefficients in this basis was given in [KM93], while the interpolation of sparse polynomials over finite fields was considered in [Man95]. It was later realized [GMS05] that this last algorithm could be considered as an approximate DFT for the special case when N is a power of two.

In the past ten or so years, a number of algorithms have appeared which directly address the problem of computing sparse approximate Fourier transforms. When comparing the results in the literature, care must be taken to identify the class of signals over which a specific algorithm is to perform, as well as to identify the error bounds of a given method. Different algorithms have been devised in different research communities, and so have varying assumptions on the underlying signals as well as different levels of acceptable error.

The first result with sub-linear runtime and sampling requirements appeared in [GGI⁺02]. They give a $\text{poly}(k, \log N, \log(1/\delta), 1/\varepsilon)$ time algorithm for finding, with probability $1 - \delta$, an approximation \hat{y} of the DFT of the input \hat{x} that is nearly optimal, in the sense that $\|\hat{x} - \hat{y}\|_2^2 \leq (1 + \varepsilon)\|\hat{x} - \hat{x}_{\text{opt}}\|_2^2$, where \hat{x}_{opt} is the best k -term approximation to \hat{x} . Here the exponent of k in the runtime is two, so the algorithm is *quadratic* in the sparsity. Moreover, the algorithm is non-adaptive in the sense that the samples used are independent of the input x . This algorithm was modified in [GMS05] to bring the dependence on k down to linear.¹ This was accomplished mainly by replacing uniform random variables (used to sample the input) by random arithmetic progressions, which allowed the use of nonequispaced fast Fourier transforms to sample from intermediate representations and to estimate the coefficients in near-linear time. The increased overhead of this procedure, however, limited the range of k for which the algorithm outperformed a standard FFT implementation [IGS07].

Around the same time, a similar algorithm was developed in the context of list decoding for proving hard-core predicates for one-way functions [AGS03]. This can be considered an extension of [KM93], and like [GGI⁺02, GMS05] is a randomized algorithm. Since the goal in this work was to give a polynomial-time algorithm for list decoding, no effort was made to optimize the dependence on k ; it stands at $k^{11/2}$, considerably higher than [GGI⁺02, GMS05]. The randomness in this algorithm is used only to construct a sample set on which norms are estimated, and in [Aka10] this set is replaced with a deterministic construction. This construction is based on the notion of ε -approximating the uniform distribution over arithmetic progressions, and relies on existing constructions of ε -biased sets of small size [Kat89, AIK⁺90]. Depending on the size of the ε -biased sets used, the sampling and runtime complexities are $O(k^4 \log^c N)$ and $O(k^6 \log^c N)$, respectively, for some $c > 4$.²

¹See [GST08] for a “user-friendly” description of the improved algorithm.

²Specifically, the runtime is $O(k^2 \cdot \log N \cdot |S|)$, where S is the set of samples read by the algorithm. This set takes the form $S = \bigcup_{\ell=1}^{\lceil \log N \rceil} A - B_\ell$, where A has ε -discrepancy on rank 2 Bohr sets, B_ℓ ε -approximates the uniform distribution on $[0, 2^\ell - 1] \cap \mathbb{Z}$, and $A - B_\ell$ is the difference set. Using constructions from [Kat89] one has $|A| = O(\varepsilon^{-1} \log^4 N)$, $|B_\ell| = O(\varepsilon^{-3} \log^4 N)$; setting $\varepsilon = \Theta(k^{-1})$ and noting that $|\bigcup A - B_\ell| = O(\sum |A - B_\ell|)$ and $|A - B_\ell| = O(|A||B_\ell|)$

In the series of works [Iwe08, Iwe10, Iwe11], a different deterministic algorithm for sparse Fourier approximation was given that relies on the combinatorial properties of *aliasing*, or collisions among frequencies in sub-sampled DFTs. By taking enough short DFTs of co-prime lengths, and employing the Chinese Remainder Theorem to reconstruct energetic frequencies from their residues modulo these sample lengths, the author is able to prove sampling and runtime bounds of $O(k^2 \log^4 N)$. The error bound is of the form $\|\hat{x} - \hat{y}\|_2 \leq \|\hat{x} - \hat{x}_{\text{opt}}\|_2 + k^{-1/2} \|\hat{x} - \hat{x}_{\text{opt}}\|_1$; it has been shown that the stronger “ ℓ_2 - ℓ_2 ” guarantee of [GMS05] cannot hold for a sub-linear, deterministic algorithm [CDD09]. Moreover, the range of k for which this algorithm is faster than the FFT is smaller in practice than that of [GMS05].

Most recently, the authors of [HIKP12b] presented a randomized algorithm that extends by an order of magnitude the range of sparsity for which it is faster than the FFT. This is accomplished by removing the iterative aspect from [GMS05] by using more efficient filters, which are nearly flat within the passband and which decay exponentially outside. In contrast, the box-car filters used in [GMS05] have a frequency response which oscillates and decays like $|\omega|^{-1}$. In addition, the identification of significant frequencies is done by direct estimation after hashing into a large number of bins rather than the binary search technique of [GMS05]. These changes give a runtime bound of $O(\log N \sqrt{Nk} \log N)$ and a somewhat stronger error bound $\|\hat{x} - \hat{y}\|_\infty^2 \leq \varepsilon k^{-1} \|\hat{x} - \hat{x}_{\text{opt}}\|_2^2 + \delta \|\hat{x}\|_1^2$ with probability $1 - 1/N$, where $\varepsilon > 0$ and $\delta = N^{-O(1)}$ is a precision parameter.

These existing algorithms generally take one of two approaches to the sparse Fourier transform problem. In [GGI⁺02, AGS03, GMS05, HIKP12b], the spectrum of the input is randomly permuted and then run through a low-pass filter to isolate and identify frequencies which carry a large fraction of the signal’s energy. This leads to randomized algorithms that fail on a non-negligible set of possible inputs. On the other hand, [Iwe10] takes advantage of the combinatorial properties of *aliasing* in order to identify the significant frequencies. This leads to a deterministic algorithm with higher runtime and sampling requirements than the randomized algorithms mentioned. Both of these randomized and deterministic approaches have drawbacks. Randomized algorithms are not suitable for failure-intolerant applications, while the process used to reconstruct significant frequencies in [Iwe10] relies on the Chinese Remainder Theorem (CRT), which is highly unstable to errors in the residues. While there do exist algorithms for “noisy Chinese Remaindering” [GRS00, Bon02, SS04] these have thus far not found application to the sparse DFT problem, and we leave this as future work.

As this paper was being prepared, the authors became aware of an independent work using very similar methods for frequency estimation in the noiseless case [HIKP12a]. Both methods consider the phase difference between Fourier samples to extract frequency information, but are based on different techniques for binning significant frequencies. The authors of [HIKP12a] use random dilations and efficient filters of [HIKP12b], whereas we use different sample lengths in the spirit of [Iwe10]. We believe both contributions are of interest, and rein-

(see, e.g., [TV06]) one obtains the stated sampling and runtime complexities.

force the notion that exploiting phase information is critical for developing fast, robust algorithms for the sparse Fourier transform problem.

1.2 New Results

In this paper we describe a simple, deterministic algorithm that avoids reconstruction with the CRT. We are thus able to avoid two pitfalls associated with existing algorithms. Our method relies on sampling the signal in the time domain at slightly shifted points, and thus it assumes access to an underlying continuous-time signal. The shifted time samples allow us to determine the value of significant frequencies in sub-sampled FFTs and also indicate when two or more frequencies have been aliased in such a sub-sampled FFT. These two key facts allow us to significantly reduce (by up to two orders of magnitude) the average-case sampling and runtime complexity of the sparse FFT over a certain class of random signals. Our worst-case bounds improve by a constant factor those of prior deterministic algorithms. We present both adaptive and non-adaptive versions of our algorithms. If the application allows samples to be acquired adaptively (that is, dependent on previous samples), we are able to improve further on our average-case bounds.

The remainder of this paper is organized as follows. In section 2 we introduce notation and prove the technical lemmas underlying our algorithms. In section 3 we introduce randomized and deterministic versions of our algorithm. In section 4 we prove that our algorithm has average-case runtime and sampling complexities of $\Theta(k \log(k))$ and $\Theta(k)$, respectively. In section 5 we present the results of an empirical evaluation of our algorithm and compare its runtime and sampling requirements to competing algorithms. Finally in section 6 we provide some concluding remarks and discuss ongoing work to appear in the future.

2 Mathematical Background

2.1 Preliminaries

Throughout this work we shall be concerned with frequency-sparse band-limited signals $S : [0, 1) \rightarrow \mathbb{C}$ of the form

$$S(t) = \sum_{j=1}^k a_j e^{2\pi i \omega_j t}, \quad (1)$$

where $\omega_j \in [-N/2, N/2) \cap \mathbb{Z}$, $a_j \in \mathbb{C}$, and $k \ll N$. The Fourier series of S is given by

$$\hat{S}(\omega) = \int_0^1 S(t) e^{-2\pi i \omega t} dt, \quad \omega \in \mathbb{Z}, \quad (2)$$

so that for signals of the form (1) we have $\hat{S}(\omega_j) = a_j$ and $\hat{S}(\omega) = 0$ for all other $\omega \in [-N/2, N/2) \cap \mathbb{Z}$. Given any finite sequence $\mathbf{S} = (s_0, s_1, \dots, s_{p-1})$ of length

p we define its Discrete Fourier transform (DFT) by

$$\hat{S}[h] = \sum_{j=0}^{p-1} s_j e^{\frac{2\pi i j h}{p}} = \sum_{j=0}^{p-1} S[j] W_p^{jh}, \quad (3)$$

where $h = 0, 1, \dots, p-1$, $S[j] := s_j$ and $W_p := e^{-\frac{2\pi i}{p}}$ is the primitive p -th root of unity. The Fast Fourier Transform (FFT) allows the computation of \hat{S} in $O(p \log p)$ steps.

We apply the DFT to discrete samples of $S(t)$ to compute the Fourier coefficients a_j of $S(t)$. For an integer p and real $\varepsilon > 0$ we form discrete arrays of samples of S of length p via

$$S_p[j] = S\left(\frac{j}{p}\right), \quad S_{p,\varepsilon}[j] = S\left(\frac{j}{p} + \varepsilon\right), \quad j = 0, 1, \dots, p-1.$$

Now assume that all $\omega_j \pmod{p}$, $1 \leq j \leq k$ are distinct. It is a simple derivation to obtain

$$\hat{S}_p[h] = \begin{cases} pa_j & h \equiv \omega_j \pmod{p} \\ 0 & \text{otherwise.} \end{cases}$$

By examining the peaks of $\hat{S}_p[h]$ we will be able to determine $\{\omega_j \pmod{p} : 1 \leq j \leq k\}$. Previous approaches applied the Chinese Remainder Theorem to reconstruct $\{\omega_j\}$ by taking a suitable number of p 's, which must overcome the problem of registrations to match up each ω_j whenever a new p is used (see e.g. [Iwe10, Iwe11]). Our algorithm takes a different approach using the shifted sub-samples. Note that

$$\hat{S}_{p,\varepsilon}[h] = \begin{cases} pa_j e^{2\pi i \varepsilon \omega_j} & h \equiv \omega_j \pmod{p} \\ 0 & \text{otherwise.} \end{cases}$$

It follows that in this setting, for $h \equiv \omega_j \pmod{p}$ we have $\frac{\hat{S}_{p,\varepsilon}[h]}{\hat{S}_p[h]} = e^{2\pi i \varepsilon \omega_j}$. Hence

$$2\pi \varepsilon \omega_j \equiv \text{Arg} \left(\frac{\hat{S}_{p,\varepsilon}[h]}{\hat{S}_p[h]} \right) \pmod{2\pi}, \quad (4)$$

where $\text{Arg}(z)$ denotes the phase angle of the complex number z in $[-\pi, \pi)$. Assume that we take $|\varepsilon| \leq \frac{1}{N}$. Then ω_j is completely determined by (4) as there will be no wrap-around aliasing, and

$$\omega_j = \frac{1}{2\pi \varepsilon} \text{Arg} \left(\frac{\hat{S}_{p,\varepsilon}[h]}{\hat{S}_p[h]} \right). \quad (5)$$

In fact, more generally, if we have an estimate of ω_j , say $|\omega_j| < \frac{L}{2}$, then by taking $|\varepsilon| \leq \frac{1}{L}$ the same reconstruction formula (5) holds. The observation that by taking slightly shifted samples will allow us to identify frequencies in $S(t)$ underlies the algorithms which follow, and the bulk of this paper analyzes various aspects of the proposed algorithms, such as efficiency and robustness.

One of the problems is that when $p < N$ it is possible that two or more distinct frequencies will have the same remainder modulo p . In this case we say the frequencies are *aliased* or *collide* (mod p). In general, for $h \in \{0, \dots, p-1\}$ and the given signal $S(t)$ let $I(S, h; p) := \{j : \omega_j \equiv h \pmod{p}\}$. Then we have

$$\hat{S}_p[h] = \sum_{\omega \equiv h \pmod{p}} \hat{S}(\omega) = p \sum_{j \in I(S, h; p)} a_j. \quad (6)$$

When aliasing occurs reconstruction via (5) is no longer valid. The aliasing phenomenon presents a serious challenge for any method with sub-linear sampling complexity. In the next section we develop a simple test to determine whether or not aliasing has occurred in a p -length DFT, which then allows us to effectively overcome this challenge and develop provably correct sub-linear algorithms.

2.2 Technical Lemmas

To effectively apply the sub-sample idea in a Fourier algorithm one must first overcome the aliasing challenge. Using shifted sub-samples gives us a simple yet extremely effective criterion to determine whether or not aliasing has occurred at a given location in a p -length DFT without resorting to complicated combinatorial techniques. Observe that complementing (6) we have

$$\hat{S}_{p, \varepsilon}[h] = p \sum_{j \in I(S, h; p)} a_j e^{2\pi i \varepsilon \omega_j}. \quad (7)$$

It follows that

$$\begin{aligned} \left| \hat{S}_{p, \varepsilon}[h] \right|^2 - \left| \hat{S}_p[h] \right|^2 &= p^2 \sum_{j, l \in I(S, h; p)} a_j \bar{a}_l e^{2\pi i \varepsilon (\omega_j - \omega_l)} \\ &\quad - p^2 \left| \sum_{j \in I(S, h; p)} a_j \right|^2. \end{aligned} \quad (8)$$

Lemma 2.1. *Let $p > 1$ and $h \in \{0, 1, \dots, p-1\}$. Assume that $q = |I(S, h; p)| > 1$, i.e. $\omega_j \equiv h \pmod{p}$ for more than one j in $S(t)$. Then we have the following:*

- (A) *Let $\varepsilon > 0$ and $E := \{\omega_j - \omega_m : j, m \in I(S, h; p)\}$. Suppose that all elements of εE are distinct (mod 1). Then $|\hat{S}_{p, m\varepsilon}[h]| \neq |\hat{S}_p[h]|$ for some $1 \leq m \leq q^2 - q$.*
- (B) *For almost all $\varepsilon > 0$ we have $|\hat{S}_{p, \varepsilon}[h]| \neq |\hat{S}_p[h]|$.*

Proof. The proof of part (B) is immediate from (8). Observe that $f(\varepsilon) := \left| \hat{S}_{p, \varepsilon}[h] \right|^2 - \left| \hat{S}_p[h] \right|^2$ is trigonometric polynomial in ε , and it is not identically 0 given that $q = |I(S, h; p)| > 1$. Thus it has at most finitely many zeros for $\varepsilon \in [0, 1)$, and hence (B) is clearly true.

We resort to the Vandermonde matrix to prove part (A). For simplicity we write $f(t) = \sum_{\alpha \in E} c_\alpha e^{2\pi i \alpha t}$. Set $r_\alpha := e^{2\pi i \alpha \varepsilon}$ where ε satisfies the hypothesis of the lemma, which implies that all r_j are distinct. Assume the claim of part (A) is false. Then we have $f(m\varepsilon) = 0$ for all $0 \leq m \leq q^2 - q$. Here $f(0) = 0$ is automatic because $\mathbf{S}_{p,0} = \mathbf{S}_p$. Thus we have

$$\sum_{\alpha \in E} c_\alpha r_\alpha^m = 0, \quad m = 0, 1, \dots, q^2 - q. \quad (9)$$

But the cardinality of E is at most $q^2 - q + 1$, which means that there are at most $q^2 - q + 1$ terms in the sum in (9). Because all r_α are distinct the matrix $[r_\alpha^m]$ is a nonsingular Vandermonde matrix, and for (9) to hold all c_α must be zero. This is clearly not the case, and a contradiction. \square

Remark. Any irrational ε or $\varepsilon = \frac{a}{b}$ with a, b coprime and $b \geq 2N$ will satisfy the hypothesis of part (A) of Lemma 2.1. It is also easy to show that in the special case where all coefficients a_j are real and $|I(S, h; p)| = 2$, we have $|\widehat{\mathbf{S}}_{p,\varepsilon}[h]| \neq |\widehat{\mathbf{S}}_p[h]|$ for any $\varepsilon = \frac{a}{b}$ with a, b coprime and $b \geq N$.

Lemma 2.1 allow us to determine whether aliasing has occurred by whether $|\widehat{\mathbf{S}}_{p,\varepsilon}[h]|/|\widehat{\mathbf{S}}_p[h]| = 1$ for a few values of ε . It offers both a deterministic (part (B)) and a random (part (A)) procedure to identify aliasing in the sub-sampled DFTs. In practice we need to set a tolerance τ in order to accept or reject frequencies according to the criterion

$$\left| \frac{|\widehat{\mathbf{S}}_{p,\varepsilon}[h]|}{|\widehat{\mathbf{S}}_p[h]|} - 1 \right| \leq \tau. \quad (10)$$

We typically choose $\varepsilon = 1/cN$ for some small constant $c \geq 2$, which would satisfy the hypothesis of part (A) of Lemma 2.1. A tolerance on the order of p/N works well in general, which is what we use in our experiments in section 5 below.

In our algorithms we will take a number of sub-sampled DFTs of an input signal $S(t)$ of the form (1), whose lengths we denote p_ℓ . Lemma 2.1 allows us to determine whether or not two or more frequencies are aliased, so that we only add the non-aliased term to our representation. Since it is unlikely that two or more frequencies are aliased modulo two different sampling rates, using a different p_ℓ in a subsequent iteration lets us quickly discover all frequencies present in $S(t)$. Lemma 2.3 gives a worst-case bound on the number of p_ℓ 's required by our deterministic algorithm to identify all k frequencies in a given Fourier-sparse signal. It is similar to [Iwe10, Lemma 1], but with a smaller constant. In its proof we use the CRT, which we quote here for completeness (see, e.g., [NZM91]).

Theorem 2.2 (Chinese Remainder Theorem). *Any integer n is uniquely specified modulo N by its remainders modulo m pairwise relatively prime numbers p_ℓ , provided $\prod_{\ell=1}^m p_\ell \geq N$.*

Lemma 2.3. *Let $M > 1$. It suffices to take $1 + (k - 1)\lfloor \log_M N \rfloor$ pairwise relatively prime p_ℓ 's with $p_\ell \geq M$ to ensure that each frequency ω_j is isolated (i.e. not aliased) $(\text{mod } p_\ell)$ for at least one ℓ .*

Proof. Assume otherwise, namely that given p_ℓ for $\ell = 1, 2, \dots, L$ with $L > k\lfloor \log_M N \rfloor$ there exists some ω_j such that ω_j is aliased $(\text{mod } p_\ell)$. By the Pigeon Hole Principle there exists at least one $\omega_m \neq \omega_j$ such that $\omega_j - \omega_m \equiv 0 \pmod{p_\ell}$ at least q times, where $q > \lfloor \log_M N \rfloor$. Without loss of generality we assume that $\omega_j - \omega_m \equiv 0 \pmod{p_\ell}$ for $\ell = 1, 2, \dots, q$. Now by the fact that $p_\ell \geq M$ we have

$$\prod_{\ell=1}^q p_\ell \geq M^q \geq N.$$

By the CRT we would then have $\omega_j \equiv \omega_m \pmod{N}$, a contradiction. \square

We remark that the algorithm in [Iwe10] requires taking $1 + 2k \log_k N$ coprime sample lengths, since that algorithm requires each ω to be isolated in at least half of the DFTs of length p_ℓ . This requirement stems from the fact that that algorithm cannot distinguish between aliased and non-aliased frequencies in a given sub-sampled DFT. Our worst-case bound is approximately a factor of two better, though in practice our algorithms never use all those sample lengths on random input. The fact that we can tell which frequencies are “good” for a given p_ℓ allows us to construct our Fourier representation one term at a time, and quit when we have achieved a prescribed stopping criterion.

3 Algorithms

Both of our algorithms proceed along a similar course; in fact they differ only in the choice of the sample lengths p_ℓ . We assume that we are given access to the continuous-time signal $S(t)$ whose Fourier coefficients we would like to determine, and further that we can sample from S at arbitrary points t in unit time. This is an appropriate model for analog signals, but not for discrete ones. In the discrete case, one could interpolate between given samples to approximate the required S -values, though we have not implemented or analyzed this case. (The same assumptions hold for the algorithms in [Iwe10], while those in [GGI⁺02, GMS05, HIKP12b] are formulated purely in the discrete realm.) In this paper mainly limit ourselves to the noiseless case. Though this is a highly unrealistic assumption, it permits a simple description of the underlying algorithm. In section 3.3 we discuss some of the problems associated with noisy signals and give a minor modification of our algorithm for low-level noise. A second manuscript in preparation addresses the issue of noise specifically, with more significant modifications to the algorithms described below.

3.1 Non-adaptive

Our algorithms start by choosing a sample length p_1 such that $p_1 \geq ck$ for some constant $c > 1$. For a fixed $\varepsilon \leq 1/N$, we then compute $\hat{\mathbf{S}}_p$ and $\hat{\mathbf{S}}_{p,\varepsilon}$,

sort the results by magnitude, and compute frequencies ω via (5) for the k largest coefficients in absolute value. We then check whether or not each of those frequencies is aliased via (6), and if it is not, we add it to our list. The coefficient is given by the unshifted sample value $\hat{\mathbf{S}}_p[h]$ at that frequency. After this, we combine terms with the same frequency and prune small coefficients from the list. We then iterate until a stopping criterion is reached. In the empirical study described in section 5, we stopped when the number of distinct frequencies in our list equalled the desired sparsity.

Our deterministic algorithm chooses p_ℓ to be the ℓ^{th} prime greater than ck . This ensures that all samples lengths are co-prime, at the expense of taking slightly more samples than necessary. By Lemma 2.3, $1 + (k-1)\lfloor \log_{ck} N \rfloor$ such p_ℓ s suffice to isolate every ω at least once. This gives us worst-case sampling and runtime complexity on the same order as [Iwe10], though the results in section 5 indicate that on average we significantly outperform those pessimistic bounds.

Our Las Vegas algorithm chooses p_ℓ uniformly at random from the interval $[c_1k, c_2k]$ for constants $1 < c_1 < c_2$. In this case we cannot make a worst-case guarantee on the number of iterations needed by the algorithm to converge. However, the results in section 5 indicate that the Las Vegas version performs similarly to the deterministic version on the class of signals tested.

3.2 Adaptive

The algorithms can also be implemented in an adaptive fashion, by which we mean that the size of the current representation is taken into account in subsequent iterations. In particular, if \mathbf{R} is our current representation, we let $k^* = k - |\mathbf{R}|$ and choose the next p_ℓ with respect to k^* instead of k . Moreover, before taking DFTs, we subtract off the contribution from the current representation, so that effort is not expended re-identifying portions of the spectrum already discovered. This idea is similar to that in [GGI⁺02, GMS05], though in our empirical studies the evaluation of the representation is done directly, rather than as an unequally-spaced FFT. This gives our algorithms asymptotically slower runtime, but the effect is negligible for the values of k studied in section 5. A formal description appears below in algorithm 1.

3.3 Modifications in the presence of noise

In the noiseless versions of the algorithms described in this paper, a test for aliasing is implemented by considering the ratio of magnitudes of shifted and unshifted peaks. When the samples are corrupted by noise, there will be two challenges. The first challenge is that the reconstruction of frequencies from shifts will be corrupted by noise. The second challenge is that there will be variations among the magnitudes even for non-aliased terms, so a higher threshold that depends on the size of the noise must be set. When this threshold is too large it affects the ability to distinguish aliased terms as there will be an increased number of false negatives. On the other hand, lower thresholds that reduce false negatives will lead to an increased number of false positives.

Algorithm 1 PHASESHIFT

Input: function pointer S , integers c_1, c_2, k, N , real ε
Output: \mathbf{R} , a sparse representation for \hat{S}
 $\mathbf{R} \leftarrow \emptyset$, $\varepsilon_0 \leftarrow 0$, $\varepsilon_1 \leftarrow \varepsilon$, $\ell \leftarrow 1$
while $|\mathbf{R}| < k$ **do**
 5: $k^* \leftarrow k - |\mathbf{R}|$ {or k if non-adaptive}
 $p_\ell \leftarrow$ first prime $\geq c_1 k^*$ {or UNIFORM($c_1 k^*, c_2 k^*$) if Las Vegas}
 for $m = 0$ to 1 **do**
 for $j = 0$ to $\ell - 1$ **do**
 $\mathbf{S}_{\ell,m}[j] \leftarrow S\left(\frac{j}{p_\ell} + \varepsilon_m\right)$
 10: $\mathbf{S}_{\text{rep}}[j] \leftarrow \sum_{(\omega, c_\omega) \in \mathbf{R}} c_\omega e^{2\pi i \omega(j/p_\ell + \varepsilon_m)}$ {omit if non-adaptive}
 end for
 $\hat{\mathbf{S}}_{\ell,m} \leftarrow \text{FFT}(\mathbf{S}_{\ell,m} - \mathbf{S}_{\text{rep}})$
 $\hat{\mathbf{S}}_{\ell,m}^{\text{sort}} \leftarrow \text{SORT}(\hat{\mathbf{S}}_{\ell,m})$
 for $j = 1$ to k^* **do**
 15: $\omega_{j,\ell} \leftarrow \frac{1}{2\pi\varepsilon} \text{Arg}\left(\frac{\hat{\mathbf{S}}_{\ell,1}^{\text{sort}}[j]}{\hat{\mathbf{S}}_{\ell,0}^{\text{sort}}[j]}\right)$
 end for
 end for
 for $j = 1$ to k^* **do**
 if $\left| \frac{|\hat{\mathbf{S}}_{\ell,0}^{\text{sort}}[j]|}{|\hat{\mathbf{S}}_{\ell,1}^{\text{sort}}[j]|} - 1 \right| < \frac{p_\ell}{N}$ **then**
 20: $\mathbf{R} \leftarrow \mathbf{R} \cup \left\{ \left(\omega_{j,\ell}, \hat{\mathbf{S}}_{\ell,0}[\omega_{j,\ell}] \right) \right\}$
 end if
 end for
 collect terms in \mathbf{R} with same ω
 prune small coefficients from \mathbf{R}
 25: $\ell \leftarrow \ell + 1$
end while

The first challenge can be addressed effectively through a combination of using larger p_j 's, multiple shifts and a multiscale unwrapping. The idea of using larger p_j 's is rather straightforward yet effective. For any given p_j the DFT detects the location of the frequencies modulo p_j rather accurately even with substantial noise. Furthermore, the reconstructed frequencies will still tend to cluster around the true value. Suppose that we sample the signal and compute DFTs of length p_j on these samples. The locations of the peaks in these short DFTs tell us the accurate value of $\omega \bmod p_j$ for each unaliased frequency ω appearing in the signal. Writing $\omega = ap_j + b$ with $a, b \in \mathbb{Z}$, we now know b and must determine a .

With a small amount of noise the reconstructed frequencies $\tilde{\omega}$ using (5) will be close to the true ω . We can thus round $\tilde{\omega}$ to the nearest integer of the form $ap_j + b$, which will recover the true frequency ω as long as $|\tilde{\omega} - \omega| < p_j/2$. For high noise levels, it is possible that the $\tilde{\omega}$ will deviate by more than $p_j/2$ from ω , so that the value for a given by rounding will be incorrect. By choosing larger p_j (i.e., increasing the parameter c_1) one can alleviate the problem somewhat, provided that the noise level is not too high. When the noise level is so high that taking a large p_j is no longer economical, a potential solution is to take multiple shifts and employ a multiscale unwrapping technique. We are still at the preliminary stage in our study of these new techniques, but early results are very encouraging.

The second challenge poses a bigger problem, but again it can be addressed in several ways. The multiscale unwrapping method will repeatedly check for aliasing at each stage, which makes it highly unlikely that an aliased frequency will pass through all the tests. Even in the unlikely event that it does, our algorithm allows false positives. Since each mode is subtracted from the original signal in our algorithm, a false positive frequency will lead to an extra mode in the new signal. As the process continues it will be extracted and cancel out the false frequency extracted earlier.

4 Average-case analysis

In this section we prove that the average-case runtime and sampling complexity of our algorithm are $\Theta(k \log k)$ and $\Theta(k)$, respectively. This is shown over a class of random signals described in section 4.2. Before giving this result on the expected runtime and sampling complexity, in section 4.1 estimate the costs of a single iteration of the while loop in algorithm 1, lines 5–25. We then describe in section 4.2 the random signal model over which we prove our average-case bounds. In section 4.3 we prove that the expected number of iterations of the while loop is constant, and in section 4.4 we use this result to prove our average-case bounds.

4.1 While loop runtime and sampling complexity

The computational cost of the while loop in algorithm 1, lines 5–25 is dominated by three operations. The first is the evaluation of the current representation \mathbf{R} of $k - k^*$ terms at the $O(k^*)$ points j/p_ℓ in line 10. In our implementation, we simply calculated this directly, looping over both the sample points and the terms in the representation. The complexity of this implementation is $O(p_\ell(k - k^*)) = O(k^*(k - k^*)) = O(k^2)$, and while non-equispaced fast Fourier transforms [DR93, AD96] yield an asymptotically faster runtime of $O(k \log(k))$, they also incur large overhead costs. For the values of k considered in this paper, the direct evaluation seems to have little effect on the overall runtime. The other two dominant computational tasks in the inner loop are the FFTs of $O(k)$ samples and the subsequent sorting of these DFT coefficients. It is well-known that both of these operations can be done in time $\Theta(k \log(k))$ [CLRS01]. Thus the inner loop has overall time complexity $\Theta(k \log(k))$, assuming the use of non-equispaced FFTs.

4.2 Random signal model

For both the average-case analysis and for the empirical evaluation described in section 5 we considered test signals with uniformly random phase over the bandwidth and coefficients chosen uniformly from the complex unit circle. In other words, given k and N , we choose k frequencies ω_j uniformly at random (without replacement) from $[-N/2, N/2) \cap \mathbb{Z}$. The corresponding Fourier coefficients a_j are of the form $e^{2\pi i \theta_j}$, where θ_j is drawn uniformly from $[0, 1)$. The signal is then given by

$$S(t) = \sum_{j=1}^k a_j e^{2\pi i \omega_j t}. \quad (11)$$

This is the standard signal model considered in previous empirical evaluations of sub-linear Fourier algorithms [IGS07, Iwe10, HIKP12b].

4.3 Markov Analysis of Collisions

In order to analyze the expected runtime and sampling complexity of our algorithms, we must estimate the expected number of collisions among frequencies modulo the sample lengths used by the algorithms. Recall that in the noiseless case, our algorithms are able to detect when a collision between two or more frequencies has occurred, and for those that are not aliased we are able to calculate the value of the frequency. Thus we seek to estimate the expected fraction of frequencies that are aliased modulo a given sample length p , since this determines how many passes the algorithm makes. In this section we derive bounds on the expected value of this quantity and discuss how the stopping criteria used in the algorithm affect its average-case performance.

In the random signal model considered in section 5, we assume the k frequencies are uniformly distributed over the bandwidth $[-N/2, N/2)$, and so the

residues $\omega \bmod p$ are also uniformly distributed over $[0, p-1]$. Our problem then becomes a classical occupancy problem: the number of collisions among the frequencies is equivalent to the number of multiple-occupancy bins when k balls are thrown uniformly at random into p bins. Define X_m to be the number of single-occupancy bins after m balls are thrown, Y_m to be the number of multiple-occupancy bins after m balls are thrown, and Z_m to be the number of zero-occupancy bins after m balls are thrown. Since p is constant, we have the trivial relationship $Z_m = p - X_m - Y_m$, so it suffices to consider only the pair (X_m, Y_m) . When the $(m+1)^{\text{st}}$ ball is thrown, we have the following possibilities:

- it lands in an unoccupied bucket, with probability $Z_m/p = 1 - (X_m + Y_m)/p$;
- it lands in a single-occupancy bucket, with probability X_m/p ;
- it lands in a multiple-occupancy bucket, with probability Y_m/p .

In the first case, we have $X_{m+1} = X_m + 1$, $Y_{m+1} = Y_m$; in the second case, we have $X_{m+1} = X_m - 1$, $Y_{m+1} = Y_m + 1$; and in the third case, we have $X_{m+1} = X_m$, $Y_{m+1} = Y_m + 1$. Conditioning on the values of X_m, Y_m we have

$$\mathbb{E} \left(\begin{bmatrix} X_{m+1} \\ Y_{m+1} \end{bmatrix} \middle| \begin{bmatrix} X_m \\ Y_m \end{bmatrix} \right) = \begin{bmatrix} 1-2/p & -1/p \\ 1/p & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad (12)$$

so that the system forms a Markov chain. By recursively conditioning on the values of X_{m-1}, Y_{m-1} , we can calculate the expected values of X_k, Y_k for any $k > 0$ using the initial condition $X_1 = 1, Y_1 = 0$. Denoting by A the matrix in the right-hand side of equation (12), we have

$$\mathbb{E} \left(\begin{bmatrix} X_k \\ Y_k \end{bmatrix} \right) = \sum_{m=0}^{k-1} \left(A^m \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) = \left(\sum_{m=0}^{k-1} A^m \right) \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \quad (13)$$

Since $\rho(A) = 1 - 1/p < 1$, where ρ is the spectral radius, the geometric matrix series can be written

$$\sum_{m=0}^{k-1} A^m = (I - A)^{-1} (I - A^k). \quad (14)$$

After some linear algebra, we obtain

$$\mathbb{E} \left(\begin{bmatrix} X_k \\ Y_k \end{bmatrix} \right) = \begin{bmatrix} k(1 - \frac{1}{p})^{k-1} \\ p(1 - (1 - \frac{1}{p})^k) - k(1 - \frac{1}{p})^{k-1} \end{bmatrix}. \quad (15)$$

Since $Z_k = p - X_k - Y_k$, we have $\mathbb{E}(Z_k) = p(1 - 1/p)^k$.

In our algorithms we choose $p = ck$ for some small integer c . Using this and the approximation $(1 + \frac{x}{n})^n \approx e^x$, we have

$$\mathbb{E} \left(\begin{bmatrix} X_k \\ Y_k \end{bmatrix} \right) \approx \begin{bmatrix} ke^{-1/c} \\ ck(1 - e^{-1/c}) - ke^{-1/c} \end{bmatrix}. \quad (16)$$

This gives a nonlinear equation for the expected number of collisions among k frequencies as a function of the parameter c . Newton's method can then be used to determine the value c required to ensure a desired fraction of the frequencies are not aliased. For example, to ensure that 90% of frequencies are isolated on average, it suffices to take $c = 5$; this value for the parameter c had already been found to give good performance in our empirical evaluation of the algorithms.

4.4 Average-case runtime and sampling complexity

In this section we will use a probabilistic recurrence relation due to Karp [Kar94, DP09] to give average-case performance bounds and concentration results for the case when the algorithm is halted after identifying k or more terms. In particular, we use the following theorem for recurrences of the form

$$T(k) = a(k) + T(H(k)), \quad (17)$$

where $T(k)$ denotes the time required to solve an instance of size k , $a(k)$ is the amount of work done on a problem of size k , and $0 \leq H(k) \leq k$ is a random variable denoting the size of the subproblem generated by the algorithm.

Theorem 4.1. [Kar94, Theorem 1.2] *Suppose $a(k)$ is nondecreasing, continuous, and strictly increasing on $\{x : a(x) > 0\}$, and that $\mathbb{E}[H(k)] \leq m(k)$ for a nondecreasing continuous function $m(k)$ such that $m(k)/k$ is also nondecreasing. Denote by $u(k)$ the solution to the deterministic recurrence*

$$u(k) = a(k) + u(m(k)). \quad (18)$$

Then for $k > 0$ and $t \in \mathbb{N}$,

$$\mathbb{P}[T(k) > u(k) + ta(k)] \leq \left(\frac{m(k)}{k} \right)^t. \quad (19)$$

Our algorithm does work $a(k) = \Theta(k \log(k))$ on input of size k and generates a subproblem whose average size is $m(k) = k/10$. (Recall from section 4.3 that with the parameter $c = 5$, on average over 90% of the frequencies were not aliased modulo $p = O(c k)$.) The associated deterministic recurrence is then

$$u(k) = \Theta(k \log(k)) + u(k/10), \quad (20)$$

whose solution is $u(k) = \Theta(k \log(k))$ (see, e.g., [CLRS01]). A straightforward application of Theorem 4.1 yields

$$\mathbb{P}[T(k) > \Theta(k \log(k)) + tk \log(k)] \leq 10^{-t}, \quad (21)$$

so that the runtime is tightly concentrated about its mean $\Theta(k \log(k))$.

The sampling complexity $S(k)$ can be handled in an analogous manner, since in this case $a(k) = \Theta(k)$ and $m(k) = k/10$ as before. The associated deterministic recurrence becomes

$$u(k) = \Theta(k) + u(k/10), \quad (22)$$

whose solution is $u(k) = \Theta(k)$. Applying Theorem 4.1 again we have

$$\mathbb{P}[S(k) > \Theta(k) + tk] \leq 10^{-t}, \quad (23)$$

so that we again have tight concentration of the number of samples around the mean $\Theta(k)$.

5 Empirical Evaluation

In this section we describe the results of an empirical evaluation of the *adaptive* deterministic and Las Vegas variants of the Phaseshift algorithm described above. Both algorithms were implemented in C++ using FFTW 3.0 [FJ05] for the FFTs, using `FFTW_ESTIMATE` plans since the sample lengths are not known in advance for the Las Vegas variant. For comparison we also ran the same tests on the four variants of GFFT as well as on AAFFT and FFTW itself. The FFTW runs utilized the `FFTW_PATIENT` plans with wisdom enabled, and so are highly optimized. The experiments were run on a single core of an Intel Xeon E5620 CPU with a clock speed of 2.4 GHz and 24 GB of RAM, running SUSE Linux with kernel 2.6.16.60-0.81.2-smp for x86_64. All code was compiled with the Intel compiler using the `-fast` optimization. As in [Iwe11], timing is reported in CPU ticks using the `cycle.h` file included with the source code for FFTW.

In the following sections we refer to our algorithm as “Phaseshift”, since by taking shifted time samples of the input signal we also shift the phase of the Fourier coefficients. To keep the plots readable, we only show data for the adaptive, deterministic variant of our algorithm; the other variants perform similarly. The algorithms of [Iwe11] are denoted GFFT-XY, where $X \in \{D, R\}$ and $Y \in \{F, S\}$. The D/R stands for deterministic or randomized, while the F/S stands for fast or slow. The fast variants use more samples but less runtime while the slow variants use fewer samples but more runtime. In the plots below, we always show the GFFT variant with the most favorable sampling or runtime complexity. Finally, AAFFT denotes the algorithm of [GMS05]. The implementations tested are summarized in table 1 along with the average-case sampling and runtime complexities, and the associated references.

5.1 Setup

Each data point in Fig. 1–2 is the average of 100 independent trials of the associated algorithm for the given values of the bandwidth N and the sparsity k . The lower and upper bars associated with each data point represent the minimum and maximum number of samples or runtime of the algorithm over the 100 test functions. The values of k tested were 2, 4, 8, \dots , 4096, while the values of N were $2^{17}, 2^{18}, \dots, 2^{26}$. For larger values of k , the slow GFFT variants and AAFFT took too long to complete on our hardware, so we only present partial data for these algorithms. Nevertheless, the trend seen in the plots

Table 1: Implementations used in the empirical evaluation.

Algorithm	R/D	Samples	Runtime	Reference
PS-Det	D	k	$k \log k$	Section 4
PS-LV	R	k	$k \log k$	Section 4
GFFT-DF	D	$k^2 \log^4 N$	$k^2 \log^4 N$	[Iwe11]
GFFT-DS	D	$k^2 \log^2 N$	$Nk \log^2 N$	[Iwe11]
GFFT-RF	R	$k \log^4 N$	$k \log^5 N$	[Iwe11]
GFFT-RS	R	$k \log^2 N$	$N \log N$	[Iwe11]
AAFFT	R	$k \log^c N$	$k \log^c N$	[GMS05]
FFTW	D	N	$N \log N$	[FJ05]

below continues for higher values of the sparsity. The test signals were generated according to the signal model described in section 4.2.

The Phaseshift and deterministic GFFT variants will always recover such signals exactly. The randomized GFFT variants are Monte Carlo algorithms, and so, when they succeed, will also recover the signal exactly. AAFFT, on the other hand, is an approximation algorithm which will fail on a non-negligible set of input signals. However, for the runs depicted in Fig. 1–2, AAFFT always produced an answer with ℓ_2 error less than 10^{-4} . The randomized GFFT variants failed a total of 7 times out of 2200 test signals, a relatively small amount that can be reduced by parameter tuning. For the Phaseshift variants, we chose the parameters $c_1 = 5$, $c_2 = 10$, and took the shift ε to be $1/2N$. Finally, for the randomized GFFT variants, we chose the Monte Carlo parameter to be 1.2.

5.2 Sampling Complexity

In Fig. 1 (a), we compare the average number of samples of the input signal S required by each algorithm when the bandwidth N fixed at 2^{22} . The sparsity of the test signal is varied from 2 to 4096 by powers of two. We can see that the Phaseshift variants require over an order of magnitude fewer samples than GFFT-RS, the GFFT variant with the lowest sampling requirements. Both Phaseshift variants also require over an order of magnitude fewer samples than AAFFT. The comparison with the deterministic GFFT variants is even starker; Phaseshift-Det requires two orders of magnitude fewer samples than GFFT-DS (not shown), and four orders of magnitude fewer samples than GFFT-DF (not shown).

In Fig. 2 (a), we compare the average number of samples of the input signal S required by each algorithm when the sparsity k is fixed at 60. The bandwidth N was varied from 2^{17} – 2^{26} by powers of two. Using powers of two for the bandwidth allows the best performance for both FFTW and AAFFT, though this fact is more relevant for the runtime comparisons in the following section. We can see that the Phaseshift variants require many fewer samples than all four GFFT variants as well as AAFFT and FFTW, for all values of N tested. The Phaseshift variants exhibit almost no dependence on the bandwidth for all

values of N , a feature not shared by the other deterministic algorithms. We note here that in future work we plan to replace the $1/2N$ shift by two or more larger shifts with co-prime denominators to obtain an equivalent shift, as in [WZ98]. This should lead to more robustness at high values of N .

5.3 Runtime Complexity

In Fig. 1 (b), we compare the average runtime of each algorithm over 100 test signals when the bandwidth N is fixed at 2^{22} . The range of sparsity k considered is the same as in section 5.2. For all values of k the Phaseshift variants are faster than GFFT-RF (the fastest GFFT variant) and AAFFT by more than an order of magnitude. When compared to GFFT-RS (not shown), GFFT-DS (not shown), and FFTW, the difference in runtime is closer to three orders of magnitude.

In Fig. 2 (b), we compare the average runtime of each algorithm over 100 test signals when the sparsity k is fixed at 60. The range of bandwidth considered is the same as in section 5.2. The Phaseshift variants are the only algorithms that outperform FFTW for all values of N tested. The other implementations tested only become competitive with the standard FFT for $N \gtrsim 2^{20}$, while ours are faster even for modest N .

5.4 Noisy Case

We report here on a preliminary study of the performance of the deterministic algorithm in the presence of noise. Our noisy signals were of the same form as in the previous section, but with complex white gaussian noise of standard deviation σ added to each measurement. As described in section 3.3, the simplest way to deal with low-level noise is to simply round the reconstructed frequencies to the nearest integer of the form $ap_j + b$, where $b \equiv \omega \bmod p_j$ is the location of the peak in a length- p_j DFT. This modification doesn't change the runtime or sampling complexity significantly, so in this section we focus on the error in the approximation as a function of the noise level σ and the parameter c_1 .

In the existing literature on the sparse Fourier transform, the ℓ_2 norm is most often used to assess the quality of approximation. There are many reasons for this choice, with the two most convincing perhaps being the completeness of the complex exponentials with respect to the ℓ_2 norm and Parseval's theorem. For certain applications, however, this choice of norm is inappropriate. For example, in wide-band spectral estimation and radar applications, one is interested in identifying a set of frequency intervals containing active Fourier modes. In this case, an estimate $\tilde{\omega}$ of the true frequency ω with $|\tilde{\omega} - \omega| \ll N$ is useful, but unless $\tilde{\omega} = \omega$ the ℓ_2 metric will report an $O(1)$ error. Furthermore, when considering non-periodic signals (equivalently, non-integer ω 's) the same precision problem appears when using the ℓ_2 metric.

For these reasons, we propose measuring the approximation error of sparse Fourier transform problems with the Earth Mover Distance (EMD) [RTG00].

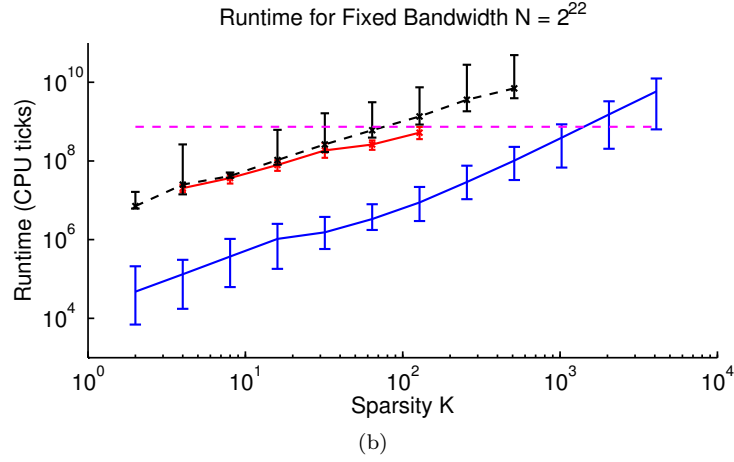
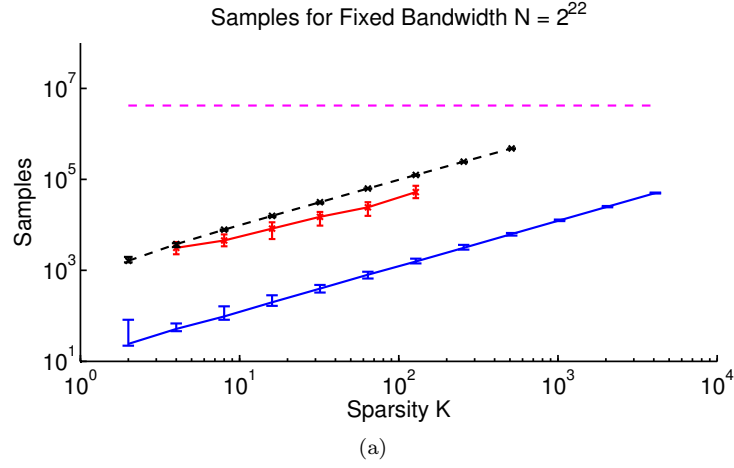


Figure 1: (a) Sampling complexity with fixed bandwidth $N = 2^{22}$ for PS-Det (blue solid line), GFFT-RS (red solid line), AAFFT (black dashed line), and FFTW (magenta dashed line). (b) Runtime complexity with fixed bandwidth $N = 2^{22}$ for PS-Det (blue solid line), GFFT-RF (red solid line), AAFFT (black dashed line), and FFTW (magenta dashed line).

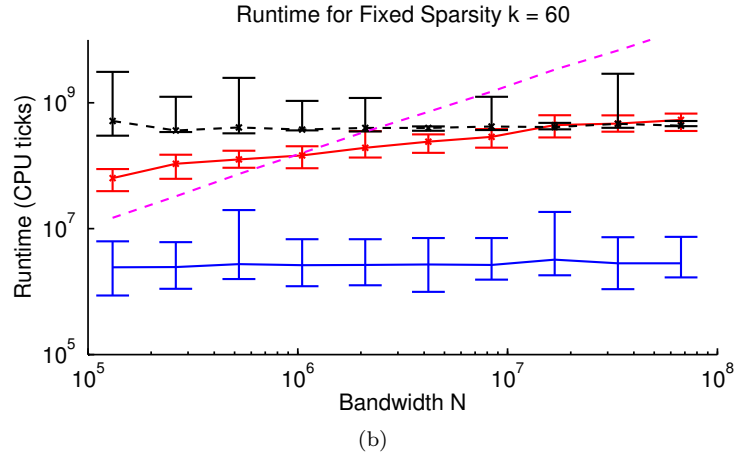
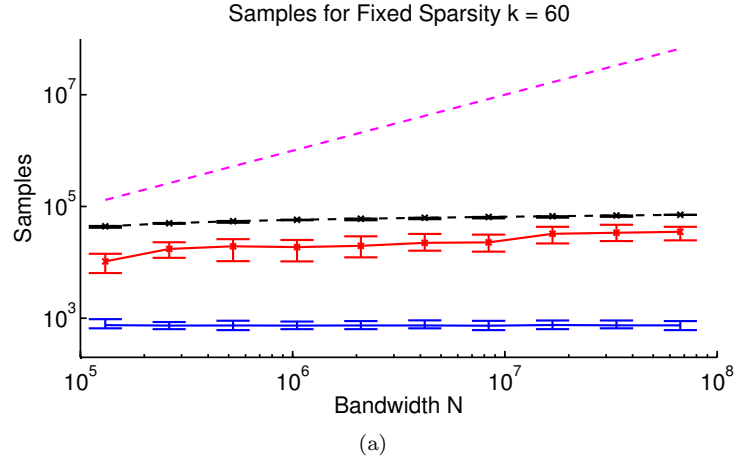


Figure 2: (a) Sampling complexity with fixed sparsity $k = 60$ for PS-Det (blue solid line), GFFT-RS (red solid line), AAFFT (black dashed line), and FFTW (magenta dashed line). (b) Runtime complexity with fixed sparsity $k = 60$ for PS-Det (blue solid line), GFFT-RF (red solid line), AAFFT (black dashed line), and FFTW (magenta dashed line).

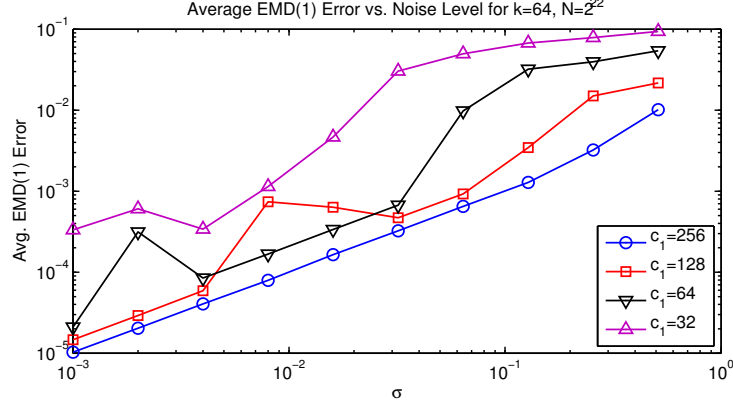


Figure 3: EMD(1) error as a function of the noise level σ for various choices of the parameter c_1 . The sparsity and bandwidth are fixed at $k = 64$, $N = 2^{22}$, respectively.

Originally developed in the context of content-based image retrieval, EMD measures the minimum cost that must be paid (with a user-specified cost function) to transform one distribution of points into another. EMD can be calculated efficiently as the solution of a linear program corresponding to a certain flow minimization problem. In our situation, we consider the cost to move a set of estimated Fourier modes and coefficients $\{(\tilde{\omega}_j, c_{\tilde{\omega}_j})\}_{j=1}^{\tilde{k}}$ to the true values $\{(\omega_i, c_{\omega_j})\}_{j=1}^k$ under the cost function

$$d_1((\omega, c_\omega), (\tilde{\omega}, c_{\tilde{\omega}}); N) \stackrel{\text{def}}{=} \frac{|\omega - \tilde{\omega}|}{N} + |c_\omega - c_{\tilde{\omega}}|. \quad (24)$$

This choice of cost function strikes a balance between the fidelity of the frequency estimate (as a fraction of the bandwidth) and that of the coefficient estimate. We denote the EMD using d_1 for the cost function by EMD(1) below.

In figure 3 we report the average EMD(1) error over 100 test signals as a function of the input noise level σ , for various choices of the parameter c_1 . In this experiment, the sparsity and bandwidth are fixed at $k = 64$ and $N = 2^{22}$, respectively. As expected, the error decreases as c_1 increases, since the rounding procedure described in section 3.3 is more likely to result in the true frequency. Moreover, the error increases linearly with the noise level, indicating the procedure's robustness in the presence of noise.

We remark that in the noiseless case the choice $c_1 = 5$ was found to be sufficient, while figure 3 indicates that the much larger value $c_1 \approx 256$ is necessary for good approximation in the EMD(1) metric. The larger sample lengths imply an increase in both the runtime and sampling complexity, and indicate that the rounding procedure of section 3.3 should be complemented by other modifications. This is the purpose of a second manuscript under preparation,

in which we combine the rounding procedure with the use of larger shifts ε_j in a multiscale approach to frequency estimation.

6 Conclusion

In this paper we have presented a deterministic and Las Vegas algorithm for the sparse Fourier transform problem that empirically outperform existing algorithms in average-case sampling and runtime complexity. While our worst-case bounds do not improve the asymptotic complexity, we are able to extend by an order of magnitude the range of sparsity for which our algorithm is faster than FFTW in the average case. The improved performance of our algorithm can be attributed to two major factors: adaptivity and ability to detect aliasing. In particular, we are able to extract more information from a small number of function samples by considering the *phase* of the DFT coefficients in addition to their magnitudes. This represents a significant improvement over the current state of the art for the sparse Fourier transform problem.

We have developed a multiresolution approach to handle the noisy case, in which we learn the value of a frequency from most to least significant bit by increasing the size of the shift ε . Finally, we are exploring the extension of these methods to handle non-integer frequencies, which would represent the first such result in the sparse Fourier transform context.

Acknowledgments

We would like to thank Mark Iwen and I. Ben Segal for making available the source code to the AAFFT and GFFT algorithms, Yossi Rubner for making available the source code for the Earth Mover Distance, and Piotr Indyk and Eric Price for sharing a preprint and source code for the sFFT algorithm. We also acknowledge helpful discussions with Anna Gilbert and Martin Strauss.

References

- [AD96] C. Anderson and M. D. Dahleh, *Rapid computation of the discrete Fourier transform*, SIAM J. Sci. Comput. **17** (1996), no. 4, 913–919.
- [AGS03] A. Akavia, S. Goldwasser, and S. Safra, *Proving hard-core predicates using list decoding*, Annual Symposium on Foundations of Computer Science, vol. 44, 2003, pp. 146–159.
- [AIK⁺90] M. Ajtai, H. Iwaniec, J. Komlós, J. Pintz, and E. Szemerédi, *Construction of a thin set with small Fourier coefficients*, Bull. London Math. Soc. **22** (1990), no. 6, 583–590.

- [Aka10] A. Akavia, *Deterministic Sparse Fourier Approximation via Fooling Arithmetic Progressions*, Conference on Learning Theory (CoLT), 2010.
- [Bon02] D. Boneh, *Finding smooth integers in short intervals using crt decoding*, Journal of Computer and System Sciences **64** (2002), no. 4, 768–784.
- [CDD09] A. Cohen, W. Dahmen, and R. DeVore, *Compressed sensing and best k -term approximation*, J. AMS **22** (2009), no. 1, 211–231.
- [CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, second ed., MIT Press, Cambridge, MA, 2001.
- [DP09] D. P. Dubhashi and A. Panconesi, *Concentration of measure for the analysis of randomized algorithms*, Cambridge University Press, Cambridge, 2009.
- [DR93] A. Dutt and V. Rokhlin, *Fast Fourier transforms for nonequispaced data*, SIAM J. Sci. Comput. **14** (1993), no. 6, 1368–1393.
- [DS00] J. Dongarra and F. Sullivan, *Guest editors’ introduction: The top 10 algorithms*, Computing in Science and Engineering (2000), 22–23.
- [FJ05] M. Frigo and S. G. Johnson, *The design and implementation of FFTW3*, Proceedings of the IEEE **93** (2005), no. 2, 216–231, Special issue on “Program Generation, Optimization, and Platform Adaptation”.
- [GGI⁺02] A. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss, *Near-optimal sparse Fourier representations via sampling*, Symposium on Theory of Computing, 2002, pp. 152–161.
- [GMS05] A. Gilbert, S. Muthukrishnan, and M. Strauss, *Improved time bounds for near-optimal sparse Fourier representations*, SPIE Wavelets XI, 2005.
- [GRS00] O. Goldreich, D. Ron, and M. Sudan, *Chinese remaindering with errors*, IEEE Transactions on Information Theory **46** (2000), no. 4, 1330–1338.
- [GST08] A. Gilbert, M. Strauss, and J. Tropp, *A tutorial on fast Fourier sampling*, IEEE Signal Processing Magazine **25** (2008), no. 2, 57–66.
- [HIKP12a] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, *Nearly optimal sparse fourier transform*, to appear in ACM Symposium on Theory of Computing (STOC), 2012.

- [HIKP12b] ———, *Simple and practical algorithms for sparse Fourier transform*, to appear in ACM-SIAM Symposium on Discrete Algorithms (SODA), 2012.
- [IGS07] M. Iwen, A. Gilbert, and M. Strauss, *Empirical evaluation of a sub-linear time sparse DFT algorithm*, Commun. Math. Sci. **5** (2007), no. 4, 981–998.
- [Iwe08] M. Iwen, *A deterministic sub-linear time sparse Fourier algorithm via non-adaptive compressed sensing methods*, Proceedings of the nineteenth annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008, pp. 20–29.
- [Iwe10] ———, *Combinatorial sublinear-time Fourier algorithms*, Found. Comput. Math. **10** (2010), no. 3, 303–338.
- [Iwe11] ———, *Improved approximation guarantees for sublinear-time Fourier algorithms*, Submitted (2011), Available at <http://www.math.duke.edu/markiwen/>.
- [Kar94] R. M. Karp, *Probabilistic recurrence relations*, J. Assoc. Comput. Mach. **41** (1994), no. 6, 1136–1150.
- [Kat89] N. Katz, *An estimate for character sums*, J. Amer. Math. Soc. **2** (1989), no. 2, 197–200.
- [KM93] E. Kushilevitz and Y. Mansour, *Learning decision trees using the Fourier spectrum*, SIAM J. Comput. **22** (1993), no. 6, 1331–1348.
- [LMN93] N. Linial, Y. Mansour, and N. Nisan, *Constant depth circuits, Fourier transform, and learnability*, J. Assoc. Comput. Mach. **40** (1993), no. 3, 607–620.
- [Man95] Y. Mansour, *Randomized interpolation and approximation of sparse polynomials*, SIAM Journal on Computing **24** (1995), no. 2, 357–368.
- [NZM91] I. Niven, H.S. Zuckerman, and H.L. Montgomery, *An introduction to the theory of numbers*, fifth ed., John Wiley & Sons Inc., New York, 1991.
- [RTG00] Y. Rubner, C. Tomasi, and L.J. Guibas, *The earth mover’s distance as a metric for image retrieval*, International Journal of Computer Vision **40** (2000), no. 2, 99–121.
- [SS04] I.E. Shparlinski and R. Steinfeld, *Noisy chinese remaindering in the Lee norm*, Journal of Complexity **20** (2004), no. 2-3, 423–437.

- [TV06] T. Tao and V. Vu, *Additive combinatorics*, Cambridge Studies in Advanced Mathematics, vol. 105, Cambridge University Press, Cambridge, 2006.
- [WZ98] Y. Wang and G. Zhou, *On the use of high-order ambiguity function for multi-component polynomial phase signals*, Signal Processing **65** (1998), no. 2, 283–296.